Research Article

A Hidden Attack Sequences Detection Method Based on Dynamic Reward Deep Deterministic Policy Gradient

Lei Zhang ¹, Zhisong Pan ¹, Yu Pan ¹, Shize Guo,¹ Yi Liu,² Shiming Xia ¹,
Qibin Zheng ³, Hongmei Li ³, and Wei Bai ¹

¹Command and Control Engineering College, Army Engineering University of PLA, Nanjing, China

²Defense Innovation Institute, Beijing, China

³Academy of Military Science, Beijing, China

Correspondence should be addressed to Wei Bai; baiwei_lgdx@126.com

Received 10 November 2021; Accepted 6 January 2022; Published 28 January 2022

Academic Editor: Marimuthu Karuppiah

Copyright © 2022 Lei Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Attacker identification from network traffic is a common practice of cyberspace security management. However, network administrators cannot cover all security equipment due to the cyberspace management cost constraints, giving attackers the chance to escape from the surveillance of network security administrators by legitimate actions and to perform the attack in both physical domain and digital domain. Therefore, we proposed a hidden attack sequence detection method based on reinforcement learning to deal with the challenge through modeling the network administrators as an intelligent agent that learns their action policy from the interaction with the cyberspace environment. Following Deep Deterministic Policy Gradient (DDPG), the intelligent agent can not only discover the hidden attackers hiding in the legitimate action sequences but also reduce the cyberspace management cost. Furthermore, a dynamic reward DDPG method was proposed to improve defense performance, which set dynamic reward depending on the hidden attack sequences steps and agent's check steps, compared to the fixed reward in common methods. Meanwhile, the method was verified in a simulated experimental cyberspace environment. Finally, the experimental results demonstrate that there are hidden attack sequences in cyberspace, and the proposed method can discover the hidden attack sequences. The dynamic reward DDPG shows superior performance in detecting hidden attackers, with a detection rate of 97.46%, which can improve the ability to discover hidden attackers and reduce the 6% cyberspace management cost compared to DDPG.

1. Introduction

In the era of cyberspace security management, it is a common practice to identify and prevent attackers from capturing, analyzing, and controlling the network traffic [1]. The possible network security management approaches based on network traffic can be divided into three categories. The first is based on the basic information, such as source address, source port, destination address, destination port, and protocol. The security equipment would check this basic information and block the flows that are not allowed. The check can be performed by the deployment of firewalls, routers, or switches, where the check rules can be access control lists, routing tables, or VLAN tags. The approach is effective and easy to deploy. But it is not flexible enough as it

can only provide access control methods based on address or service. The second approach is to extract the characteristic information from a load of network traffic and map it to the high-level semantics to realize the identification of attackers. Currently, Intrusion Detection System (IDS) [2,3] and Intrusion Prevention System (IPS) [4,5] are widely applied to detect attackers in this way. This method is essentially an extraction process of a single data packet or message sequence characteristics, and it is easy to be extended by machine learning algorithms. Recently, academia and industry are also putting forward a lot of effective algorithms and automating the malicious traffic feature extraction step by step, usually to judge malicious traffic on a single data or a single data stream and the lack of understanding of the entire cyberspace security situation. The third approach is to

capture, store, and analyze the network traffic centrally from multiple network links [6]. The typical products emerging in recent years are named security situational sense, which are able to use the information from more than one link and improve the identification precision of the multistep attack or coordinated attack. However, the effectiveness of the approach is largely dependent on the integrity of data stream collection. If there are not adequately monitored network links, it is difficult to identify hidden attackers accurately or precept the entire cyberspace security situation accurately.

In the actual cyberspace security management, considering the cost of equipment budget and cyberspace management, it is impossible to collect and store the entire network traffic. Only several key links are selected to be monitored to achieve part of the basic data capture and analysis. It is difficult to detect the hidden attacks effectively using the traditional detection algorithm based on traffic characteristics of the temporal sequence of the users, as the detailed user temporal sequence analysis is absent. Thus, the attacker may take legitimate action sequences to bypass the surveillance of security equipment and perform attacks. This paper simulated a typical cyberspace environment and proposes a method of attacker detection policy based on reinforcement learning. This method can analyze the possibility of attack based on the observed user's action sequences. According to the reward information obtained by taking different actions in different states, the security protection policy of administrators is generated through self-learning to effectively avoid the problem that attacker detection relies too much on the integrity of collected data and reduce cyberspace management costs.

The main contributions of this paper are as follows:

- (i) The possibility of bypassing the security device was analyzed and discussed, and then the attack was simulated. Furthermore, a corresponding simulation cyberspace environment is constructed. The simulated cyberspace environment showed that the attacker can bypass the security device to carry out an attack by hiding in the legitimate action sequences.
- (ii) A hidden attack sequence detection method was proposed based on DDPG to automatically generate a hidden attacker detection policy for network administrators through learning from the reward obtained from different actions taken by the agent.
- (iii) A dynamic reward DDPG method was proposed to detect the hidden attacker who is using a hidden attack sequence launch attack, which performed better in discovering the hidden attacker and reducing cyberspace management cost than the common DDPG method.
- (iv) The experimental results showed that the proposed method based on DDPG could effectively detect the hidden attack covered by the legitimate action sequences, and the dynamic reward DDPG method is better than the DDPG method in discovering attackers and reducing the cyberspace management cost.

The rest of this paper is organized as follows: In the next section, we introduce the related work, including intrusion detection and reinforcement learning. An intelligent hidden attacker detection model is proposed in Section 3. Experiments and discussions are proposed in Section 4. Finally, we summarize the paper in Section 5.

2. Related Work

2.1. Intrusion Detection. Intrusion detection plays an important role in cyberspace security protection [7]. There have been many great achievements in the past decades. The IDS or IPS has become a fundamental and necessary security device in modern cyberspace security management. There are many kinds of IDS. Based on the location, they can be divided into host-based IDS and network-based IDS, the main object of the host-based IDS is the system's behaviors of the target program [8], and the main object of the network-based IDS is the analysis of network traffic. According to different ways of finding intrusions, IDS can be divided into feature-based, anomaly-based, and hybrid-based [9]. Feature-based IDS is based on the predefined network attack mode of network security experts.

It compares program action with network traffic action to identify intrusion action. It has the advantages of high accuracy and easy promotion. However, it lacks the detection of unknown attacks. The anomaly-based IDS, by learning the legitimate behavior patterns of the network, discovers abnormal behaviors and traffic and has a stronger ability to detect unknown attacks but has a higher false-positive rate than feature-based IDS. The hybrid-based IDS combines the advantages of both.

At present, the network-based IDS is widely concerned and applied due to the advantages of no perception of the terminal and no occupation of host resources. In the extraction of malicious traffic features, the traditional methods mainly rely on manually extracting the characteristics of known malicious traffic. In recent years, the emergence of machine learning has made automatic learning network traffic characteristics become a new trend. There are many new attempts, including SVM [10,11]; K-nearest neighbors [12]; Naive Bayes [13]; random forests [8]; neural network [14]; and deep learning. The methods based on deep learning have become mainstream in the field because of their better performance. An IDS architecture based on a deep belief network, which uses a multilayer unsupervised learning network (restricted Boltzmann machine) and a supervisor-based backpropagation network, is proposed [15] and verified on the KDD99 dataset. At the same time, an asymmetric depth self-encoder to learn network traffic characteristics is unsupervised and achieves good performance on large sample data sets while reducing training time [16].

In another way, the IDS model used the RNN model, then compared the effectiveness of the nondepth model with the NSL-KDD dataset, and achieved good performance [17]. An IDS model using LSTM and gradient descent strategy was proposed. The performance of the model was compared in terms of accuracy, detection rate, and false-positive rate, which proved the effectiveness of the LSTM in IDS [18]. A

comprehensive study on the anomaly-based and deep learning IDS model compared the comprehensive performance of RNN, LSTM, and self-encoder on the NSL-KDD dataset and proved that the deep learning method can not only be used in the field of intrusion detection but also achieve better performance [19]. In order to mitigate the inconsistency between dimensionality reduction and feature retention in imbalanced IBD, a variational long short-term memory learning model for intelligent anomaly detection based on reconstructed feature representation is proposed. An encoder-decoder neural network associated with a variational reparameterization scheme is designed to learn the low-dimensional feature representation from high-dimensional raw data [20].

Most of the above-mentioned intrusion detection methods of network traffic are supervised, mainly by using some established datasets to train them, such as KDD99, NSL-KDD, WIDE, and AWID datasets. The process is very complicated, and the trained model is difficult to find out the unknown threats. Therefore, some scholars turn their attention to reinforcement learning. The intrusion detection method based on reinforcement learning no longer needs a training dataset. The intrusion detection agent and the environment interact with each other, continuously improving the performance of the intrusion detection agent; it can not only use the traffic for input but also use the various actions of the network for input. The application of reinforcement learning to the classification of data streams proves that it has similar effects on the SVM [21]. Researcher constructed a small simulation environment to illustrate the use of reinforcement learning to detect network malicious traffic, but the environment and method are very simple, and it is difficult to handle the needs of a complex environment [22]. A multi-agent-based DDOS traffic attack protection method is proposed, whose main goal is to ensure that the target host is not crashed by optimizing the number of discarded data packets [23]. Both the attacker and the defense can dynamically alter their security rules in a simulated model in which the heart-bleed environment exists [24].

In the game model, the zero-sum game model is under incomplete information on cyberspace, in which both the attacker and the defender attempt to win the game, and this game process cannot be described by the classifier [25]. Also, a method that combined reinforcement learning and supervised learning was proposed and then applied to the malicious traffic detection model to achieve the integration of reinforcement learning and supervised learning and achieved better performance [26]. A survey of DRL approaches developed for cybersecurity was published. The survey touches on different vital aspects, including DRL-based security methods for cyberphysical systems, autonomous intrusion detection techniques, and multiagent DRL-based game theory simulations for defense strategies against cyberattacks [27].

2.2. Reinforcement Learning. Reinforcement learning is commonly considered as a general artificial intelligence model, and it mainly studies how an agent can learn a certain

policy by interacting with the environment to maximize long-term rewards. Reinforcement learning is based on the Markov Decision Process (MDP) [28]. An MDP is a tuple (S, A, T, R, γ) , where S is the set of states and A is the set of actions. $T(s_i|s_j, a): T \times A \rightarrow R$ is the reward after executing action a at state s_i , and γ is the discounting factor. We used π to denote a stochastic policy, $\pi(s, a): T \times A \rightarrow [0, 1]$ is the probability of executing action a at state s , and $\sum_{a \in A} \pi(s, a) = 1$ for any s . The goal of reinforcement learning is to find a policy π that maximizes the expected long-term reward. Besides, the state-action value function is

$$Q^\pi(s, a) = E \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_t \sim \pi(s_t) \right], \quad (1)$$

where $\gamma \in (0, 1]$ measures the importance of future reward to current decisions.

For different policies π , they represent the possibility of different actions selected in the same state and also correspond to different rewards. A better policy can select better action in the same state to obtain more rewards. The whole process is shown in Figure 1.

In traditional reinforcement learning, the action-value function is calculated interactively and will eventually converge and obtain the optimal policy, mainly including Q-learning, Monte-Carlo learning, and temporal-difference learning. After deep learning was proposed, the deep reinforcement learning method formed by combining reinforcement learning with deep learning is the mainstream method at present.

In the following, we introduce two categories of methods based on action-value DQN and policy gradient DDPG:

Q-Learning. Q-learning [29] is a famous method based on the action-value function. It is one of the main reinforcement learning algorithms, and it is a model-free learning method. A key assumption on which Q-learning is based is that the interaction between the agent and the environment can be seen as an MDP. Q-learning is based on a fixed state transition probability distribution, the next state, and an immediate reward. In Q-learning, each $Q(s, a)$ corresponds to a Q-value; in the learning process, actions are selected according to Q-value. Q is defined as the sum of the rewards you will get if you execute the current action and follow through on a particular policy. It is defined as follows:

$$Q^*(s, a) = \gamma \sum T(s, a, s') \max_{a'} Q^*(s', a') + r(s, a), \quad (2)$$

where s is the set of states, a is the set of actions, $T(s, a, s')$ is the probability of performing the action a in state s and transition to state s' , $r(s, a)$ is the reward for executing action a in state s , and γ is the discount factor, which means the importance of future reward for the current action. In the process of learning, actions are selected according to the Q-value, and the Q-value is adjusted according to the reward of actions to gradually make the reasonable actions correspond to a higher Q-value and make them more likely to be selected in the subsequent actions. Q-learning has

FIGURE 1: The basic process of reinforcement learning.

received extensive attention on traditional reinforcement learning due to its simplicity and feasibility. However, if there are many states and actions, the Q-table will be very large, and the learning convergence process will be slow. Therefore, the deep Q-network is proposed.

Deep Q-Network (DQN). Mnih et al. [30] combined a neural network with the Q-learning algorithm in common reinforcement learning, and the DQN is proposed.

Instead of recording Q-value with Q-table, DQN uses a neural network to predict Q-value and constantly updates the neural network to learn the optimal policy. There are two neural networks in DQN. One is a relatively fixed network, called target network, which is used to obtain the value of Q-target; the other is evaluate-network, which is used to get the value of Q-evaluate. Formally, DQN minimizes the following loss function:

$$L(w) = E \left[\left(R + \gamma \max_{a'} Q(s', a'; \theta_t) - Q(s, a; \theta) \right)^2 \right]. \quad (3)$$

$Q(s, a; \theta)$ represents the output of the current network and is used to evaluate the value function of the current state action; it is the output of the current evaluate-network and is used to evaluate the value function of the current state action. $Q(s', a'; \theta_t)$ is the target network, the current network parameter θ is updated in real time, and the parameters are updated by minimizing the mean square error between the current Q-value and the target Q-value and use Stochastic Gradient Descent (SGD) to update network parameters. After the update of the target value network, the target Q-value remains unchanged for a period, which reduces the correlation between the current Q-value and the target Q-value to some extent, and improves the stability of the algorithm.

DQN is based on value functions. When actions are continuous actions, it is impossible to enumerate the benefits of all actions to determine the action with the greatest benefits. For this reason, Silver et al. [31] proposed the DPG method so that when adjusting policies at each step, it is not necessary to calculate the benefits of all policies and select the optimal policy, but it only needs to adjust the parameters of policies along the direction that makes the objective reward larger.

Thus, the blindness of policy selection is greatly reduced, and the algorithm can quickly converge to a better policy. In this process, the policy gradient used is as follows:

$$\nabla_{\theta} J(\pi_{\theta}) = E_{s \sim \rho^{\pi}, a \sim \pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a | s) Q^{\pi}(s, a)]. \quad (4)$$

In order to calculate $Q^{\pi}(s, a)$, the actor-critic algorithm is also introduced to DPG, which includes an actor network and a critic network. The actor network adjusts parameters according to equation (4) to select the action, while the critic network estimates $Q^{\pi}(s, a)$, and the critic network adjusts its parameters by minimizing the mean square error, constantly approximating the real value.

Deep Deterministic Policy Gradient (DDPG). DDPG [32] is a learning method that integrates deep learning neural network into DPG. Compared with DPG, the improvement of the use of the neural network as a policy network and Q-network then used deep learning to train the above neural network. DDPG has four networks: actor current network, actor target network, critic current network, and critic target network. In addition to the four networks, DDPG also uses experience playback, which is used to calculate the target Q-value. In DQN, we are copying the parameters of the current Q-network directly to the target Q-network, that is, $\theta^Q = \theta^Q$, but DDPG uses the soft update. And this is the loss function:

$$L(w) = \frac{1}{m} \sum_{j=1}^m (y_j - Q(\phi(S_j), A_j, w))^2, \quad (5)$$

$$\begin{cases} \theta^Q \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}, \\ \theta^{\mu} \leftarrow \tau \theta^{\mu} + (1 - \tau) \theta^{\mu'}, \end{cases}$$

where τ is the update coefficient, which is usually set as a small value, such as 0.1 or 0.01. And this is the loss function:

$$L(w) = \frac{1}{m} \sum_{j=1}^m (y_j - Q(\phi(S_j), A_j, w))^2. \quad (6)$$

For example, in the same state, we have two different output actions a_2 and a_1 and get two pieces of feedback from critic current network of Q-value, respectively, Q_1 and Q_2 , assuming $Q_1 > Q_2$, and then the agent will take action a_1 to get more awards, so the policy is increasing the probability of a_1 and reducing the probability of a_2 ; that is, the actor wants as far as possible more Q-value.

3. Hidden Attack Sequences Detection Method

In this section, we proposed a hidden attack sequences detection method based on DDPG. First of all, we simulated a typical cyberspace environment, and it was proved that the attacker can bypass the security device to launch an attack covered by legitimate action sequences. Based on this, a

hidden sequence detection method based on DDPG was proposed. Finally, we proposed a dynamic reward DDPG method to improve the performance. The following will describe the basic model and give the relevant details.

3.1. Typical Cyberspace Environment. The typical cyberspace environment is derived from the real cyberspace of enterprise cyberspace. The enterprise network is mainly divided into two parts: a business network and a management network. The business network is mainly for internal users to access various business systems. The management network is mainly for network administrators and configuration of the network devices. The business network and the management network cannot communicate with each other. In the business network and the management network, terminals, switches, servers, and security protection devices are configured, respectively. The simplified cyberspace is shown in Figure 2.

In this cyberspace environment, the business network contains six devices, including one terminal (Terminal-1), three servers (Server-1, Server-2, and Server-3), one switch (Switch-1), and one firewall (Firewall). Due to business security needs, the following security policy is set on the Firewall: Terminal-1 is allowed to access information services on Server-2 or Server-3, but it is not allowed to access security information on Server-1. Terminal-1 has the right to access Server-2 and Server-3 through its remote desktop services. Mutual access among servers is prohibited. In the management network, four devices are involved, including one terminal (Terminal-2), one switch (Switch-2), one server (Server-4), and one Intrusion Prevention System. Through the management network, Terminal-2 can access Server-4 by remote desktop service. The Intrusion Prevention System monitors the traffic from Terminal-1 to Firewall and Terminal-2 to Switch-2. The monitoring information is mainly based on the source address, the destination address, the source port, the destination port, and the destination service. For abnormal traffic, an alarm is issued.

In the current security configuration, the hidden attacker can access the security information in Server-1 through some carefully constructed hidden attack sequences. The hidden attack sequence is as follows: first, the attacker uses Terminal-2 to access Server-4 by remote desktop service; second, the attacker accesses Firewall through Server-4, modifying the access control list and allowing Server-2 to access Server-1; third, the attacker uses Terminal-1 to access Server-2 by remote desktop service and then uses Server-2 to access the security information of Server-1; finally, the attacker can use Terminal-2 to access Server-4 and then access the firewall management service by Server-4 to delete the access information to complete the attack.

During this attack, the Intrusion Prevention System monitors the link “Terminal-1 to Firewall” and link “Terminal-2 to Switch-2” at the same time but only monitors Terminal-1 accessing Server-2 and Terminal-2 accessing Server-4, so it will not alarm, but the attacker has actually completed the attack. In this cyberspace, there are many similar hidden attack sequences; this puts forward challenges

FIGURE 2: Typical cyberspace environment.

to administrators and cyberspace security management systems.

3.2. Model Design. Cyberspace security management is a typical game process between attackers and administrators, which is suitable for the framework of reinforcement learning.

Reinforcement learning is a generic model, which has a wide range in intrusion detection. The administrators will not only focus on the characteristics of single network traffic but also pay more attention to the relationship between different network traffic in their daily network management, which means the combination of generic intrusion detection knowledge and specific network security features. Those network security features are learned automatically by interacting with the network environment constantly and getting the rewards. The combination will lead to a security management strategy, which is more suitable to the network as well as less network management cost. Based on these considerations, this paper proposes an attacker intelligence detection model based on reinforcement learning. Based on the DDPG, the model can intelligently generate network security management policies applicable to the network based on analysis of the continuous feedback with the network administrator. The attacker intelligent detection model can discover the hidden attack sequence steps and reduce the cost of network management.

The basic model of the hidden attacker intelligent detection model is shown in Figure 3. The model is divided into three modules, including intelligent analysis engine module, network state awareness module, and multidomain action execution module. The intelligent analysis engine module is the core part, which is mainly responsible for choosing a proper action based on the current network states. The network state sense module is mainly responsible for getting the current network state which is based on the sensors deployed previously. As mentioned above, due to cyberspace management cost constraints, the observed cyberspace state

FIGURE 3: Hidden attack sequences detection model.

is only one part of the overall network state. The main function of the multidomain action execution module is to perform multidomain actions and to get corresponding rewards. This module can not only perform some network actions but also perform some physical domain action and information domain action, which also means that this module can act as a software module, a camera, a sensor, or other entities. As long as it can perform a specific action and perceive the corresponding reward, it can be used as the module.

As can be seen from the above analysis, the model can not only protect against attacks from the network but also be used for attacks from physical domains, network domains, and social domains as long as they can meet some simple prerequisites. In this paper, we called this network that includes physical domain, cognitive domain, and social domain cyberspace. These prerequisites include the following:

- (i) Attacks should be independent and identically distributed. In a cyberspace environment, the attacks should be independent; that is, there is no dependency between the two attacks, and the probability of occurrence of various attacks is roughly equal. For a real cyberspace environment, it is often to face many organizations and different types of attackers, there is no synergy between these attackers, and the attack capabilities can be roughly divided into several levels and for common types and means of attacks can be roughly considered meeting the requirements of independent and identical distribution.
- (ii) The reward of multidomain actions can be measured. Using this model for the detection of attackers, another necessary condition is that the reward of multidomain actions can be measured, and this metric should be a simple scalar. For a real cyberspace environment, with the security management department of the network, the reward of a specific multidomain action can be quickly evaluated and measured, which makes the attackers intelligent detection model not only be able to quickly learn network security

management online but also have the ability to respond quickly to the changes in the network.

- (iii) The cyberspace state should be sensation. The third requirement for using this model is the need to sense the state of the cyberspace, which is the primary input to the model; the intelligent analysis engine module analyzes, evaluates, and selects the appropriate actions based on these inputs. For intrusions in different domains of the cyberspace, the sensed security state is different. It may be the state of the personnel in the physical domain entering or leaving the space, the action of the computer network in the network domain, or the reading and writing of information. The collection of these states is a prerequisite for judging the attacker.

3.3. Intelligent Analysis Module. The core of the model is the intelligent analysis module, which is a DDPG model. The model followed the standard operation procedures, including sensing the cyberspace environment, performing corresponding actions and getting rewards, and then further training of the network. The network adopts the DDPG, and its main structure is shown in Figure 4.

The intelligent analysis module mainly consists of four networks and one experience playback memory. Among the four networks, there are two policy networks (Actor) and two Q-networks (Critic), which are an online policy network, a target policy network, an online Q-network, and a target Q-network.

The two policy networks have the same structure as shown in Figure 5. The input is the state of the cyberspace, and the output is the action that needs to be selected. Structurally, a layer of RNN hidden nodes is added between the input layer and the hidden layer of the DDPG. The policy network is divided into five layers: the first layer is the input layer; the second layer is the RNN hidden layer, which contains 32 GRU structure nodes; the third layer and the fourth layer are fully connected layers, including 48 fully connected nodes and activation function using the ReLU function; the last layer is the output layer, using the Sigmoid function as the activation function and finally generating a

FIGURE 4: Intelligent analysis module A standard DDPG algorithm model.

FIGURE 5: Policy network structure.

FIGURE 6: Q-network structure.

multidimensional vector representing the multidomain action that needs to be performed.

The two Q-networks have another structure, as shown in Figure 6. The input not only is the state of the cyberspace but also includes a multidimensional vector, representing the corresponding multidomain action, and the output is a scalar, representing the state and action corresponding Q-value. The network is divided into four layers: the first layer is the input layer; the second layer and the third layer, respectively, contain 48 fully connected nodes; the last layer is the output layer and the activation function using the ReLU function and outputs a scalar, using a linear function as the activation function to represent the state and action corresponding Q-value.

3.4. Dynamic Reward DDPG. The whole intelligent analysis model is mainly based on DDPG. According to the input of the cyberspace state sense module and the

feedback of the cyberspace environment, the four networks in the intelligent analysis module are optimized and adjusted in real time to generate hidden attacker detection action.

The setting of reward has a great influence on the learning effect of DDPG. Since the basic assumption of reinforcement learning is to maximize cumulative rewards, the design of rewards must be a critical step. In general DDPG, the reward is also a fixed value. Generally, the reward setting in the simulated environment is that if an attack is successful, the agent will get a negative reward, like -100 , and if the attacker is captured by the agent, the agent will get a positive reward, like 100 . In order to improve the efficiency of the method, a dynamic reward DDPG was proposed in which reward is no longer a fixed value, and the reward R is updated following this equation

$$R = \frac{R_c + R_s}{n_f} + \frac{C}{na_{\text{step}}} - R_{\text{cost}} \times n_{\text{step}}. \quad (7)$$

R_c is the agent's reward when the attacker is captured, R_s is the agent's reward when the attacker is attacked successfully, n_f is the number of attacks using one same path, na_{step} is the attack sequence steps, and n_{step} is the agent's check cyberspace steps. In addition, we set a R_{cost} to represent the cost of checking the cyberspace whether it is attacked, so we set a negative reward to the agent. C is a constant term, which makes sure that R does not change dramatically, and its value will be obtained from experiences and experiments.

Based on the above description, the agent reward is set following this equation

$$\left\{ \begin{array}{l} R = \frac{R_c}{n_f} + \frac{C}{na_{\text{step}}} - R_{\text{cost}} \times n_{\text{step}}, \\ \text{if attacker captured,} \\ R = \frac{R_s}{n_f} - R_{\text{cost}} \times n_{\text{step}}, \\ \text{if attacker attack successfully,} \\ R = -R_{\text{cost}} \times n_{\text{step}} \\ \text{if agent check cyberspace but no harvest,} \\ R = 0 \text{ otherwise.} \end{array} \right. \quad (8)$$

Compared with the common reward setting, the present method has the following advantages:

- (i) A parameter R_{cost} was introduced in this reward setting. The parameter means that if the agent keeps checking the cyberspace, it will need maintenance cost. In general, the agent can find the attacker if it checks every moment, but it will cost high cyberspace management costs, so it is not reasonable.
- (ii) A parameter n_{step} was introduced in this reward setting. As we know, the possibility of capturing the attacker increases with the length of the hidden attack sequences. Conversely, the shorter length of the hidden attack sequences increases the difficulty for the agent to capture the attacker.
- (iii) In this cyberspace environment, there are many hidden attack paths; for the agent, a large positive reward was presented to the agent when the hidden attack was captured in one path for the first time, while the reward would decrease with the detection in the same hidden attack path. Consequently, it

would encourage the agent to detect more different hidden attack paths so as to improve the detection performance of the agent.

Generally, reward setting is a design method that relies heavily on experiments and experiences. However, we just put forward a better reward setting method, and there must be a better reward setting method. One of the problems with reinforcement learning is that it is difficult to set rewards. It is possible that the program is logically correct, but the experiment does not converge, or the proposed method does not improve. Therefore, designing a reward depends on a certain amount of experience and continuous trying.

3.5. Process of the Proposed Method. In a word, the main steps include the following:

- (1) Initialize each module of the intelligent analysis engine, including randomly initializing the online Q-network $Q(s, a | \theta^Q)$ and the online policy network $\mu(s | \theta^\mu)$, initialize the target policy network μ' and the target Q-network Q' using parameters of the online Q-network and the online policy network, that is $\theta^Q \leftarrow \theta^Q$ and $\theta^\mu \leftarrow \theta^\mu$, and initialize the experience playback memory which is empty.
- (2) Uninterruptedly obtain the current state of the cyberspace from the network state sense module, assuming that the state s_t of its input is at time t .
- (3) Using the online policy network, select the corresponding action $\mu(s_t)$ according to the input state, and add a certain noise according to the action, so that the model can acquire a certain exploration ability. Call the multidomain action execution module to perform the action and get the corresponding reward R_t .
- (4) The state of the next time s_{t+1} is obtained through the network state sense module, and then the quad (s_t, a_t, r_t, s_{t+1}) is stored to the experience playback memory.
- (5) Randomly select N random state transition sequences $N^*(s_i, a_i, r_i, s_{i+1})$ from the experience playback memory, input the target strategy network and the target Q-network, calculate $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1} | \theta^{\mu'})) | \theta^{Q'}$, and calculate the loss:
$$L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i | \theta^Q))^2. \quad (9)$$
- (6) Using the gradient descent, the online Q-network is updated under the condition of minimizing loss L .
- (7) Update the online policy network by using the sampling policy gradient.

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a | \theta^Q) \bigg| \nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a | \theta^Q) \bigg|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s | \theta^\mu) \bigg|_{s_i}. \quad (10)$$

- (8) Update the target policy network and the target Q-network by using the updated online policy network and the online Q-network. In this process, τ is generally taking a smaller value, such as 0.01.

$$\begin{cases} \theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}, \\ \theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}. \end{cases} \quad (11)$$

4. Experiments

4.1. Experiment Datasets. In this section, we present experiments to support our proposed method. The python program is used to simulate the cyberspace environment shown in Figure 7, and the corresponding users' actions are obtained. Then, the model proposed in this paper is implemented, and the corresponding experiments are carried out. In this experiment, the intelligent module obtains the network actions of the corresponding users by observing the monitored target traffic and judges whether the user is an attacker according to it. If it is an attacker, the agent will capture the attacker. Otherwise, the agent will not check the cyberspace to save cyberspace management costs. Therefore, in this experiment, when the cyberspace state is input to the intelligent analysis module, the intelligent analysis module will output an action, and then the agent will perform an action. According to the experimental setup, the agent has two actions: check cyberspace or do not check.

During the experiment, users located in the building are randomly generated. Each user identifies whether it is an attacker when it is generated. In the entire user group, the proportion of hidden attackers is UP . If a user is a normal user, it is only in the randomly selected actions in network actions that can be performed normally, each user performs a certain number of actions in two rooms, each action occupies one time slice, and users who exceed the time slice will be forced to exit by the system if they do not exit by themselves. If a user is an attacker, it randomly launches an attack, and the probability of launching the attack at each step is AP . When the attacker starts the attack, he tends to execute the complete attack sequences steps as soon as possible. If he succeeds, he will exit the room by himself. If an attacker is captured during the execution of the attack sequence, he will be forced to exit the cyberspace environment.

The network administrator can get the corresponding reward through the current action. When the administrator finds that the server is attacked and captures the attacker, he gets a certain reward R . When the administrator checks the server status, but the server is not attacked, he gets a certain negative reward RC . When the administrator does not check the server, but the server is successfully attacked, the administrator gets more negative reward RP .

In the experiment, each time slice the user performs an action, the agent performs an action and gets the corresponding reward. During the model training process, rewards for all actions of agents were added as a reward for training. The model will be trained 500 times during each training session. The other parameters in the experiment are set as follows: in the experimental cyberspace environment, the attacker's ratio is $UP=0.4$, the attacker's ratio of launch attack is $AP=0.3$, and the number of actions performed by each user is no more than 60. In each training or testing, 500 users are generated.

4.2. Baselines. In order to prove the performance of the proposed method, the baselines are as follows:

- (1) *Random Check.* The first method is that the administrator randomly checks whether there is an attack currently. In this method, a parameter γ is introduced, which takes a value range of $[0, 1]$, which represents the proportion of not checking cyberspace in the total agent's actions. $\gamma=0$ represents that the agent will never check the cyberspace.
- (2) *Packet Detection.* In traditional malicious traffic identification, deep packet detection or supervised learning is usually used to extract the features of malicious traffic, which can be regarded as malicious detection based on some data packets' features. Since the corresponding traffic is not generated in the simulation cyberspace, the simulation detection strategy is adopted to make the approximation; that is, when the agent finds that the user is connecting to Server-2 or Server-4 through the remote desktop, he will check whether the server is attacked and catch the attacker.
- (3) *DQN.* When using DQN, the parameters are set as follows: learning rate is 0.01, reward discount coefficient is 0.9, exploration probability is 0.1, target network replacement iteration number is 200, and memory upper limit size is 2000.
- (4) *DDPG.* When using DDPG, in the reward part, we set $R=100$, $RP=-100$, and R_{cost} takes a value between 0 and 20 following the DRDDPG method. The discount factor $\beta=0.99$, the actor network learning rate is $1e-4$, and the critic network learning rate is $1e-3$.
- (5) *Dynamic Reward DDPG Method (DRDDPG).* This method is proposed in this paper. In the reward setting and from parameters R_c , R_s , R_{cost} , and C , we will select the best parameters by experiments. At last, other parameters are consistent with DDPG.

We use the following indexes as the evaluation criterion: the reward that the agent finally obtained, the number of

FIGURE 7: Reward value during DDPG training: the x -axis is the episode number and the y -axis is the total reward the agent obtains in an episode. (a) $R_{\text{cost}} = 5$. (b) $R_{\text{cost}} = 10$. (c) $R_{\text{cost}} = 15$. (d) $R_{\text{cost}} = 20$.

attackers that successfully attacked Server-1, the number of captured attackers, and detection rate. When the reward and detection rate is higher, the number of attackers successfully attacked is smaller, the number of captured attackers is more, and the better of the method will be proved. Also, the cyberspace management cost is hard to measure; in this paper, we use reward to measure cyberspace management cost, that is when reward is higher, the cyberspace management cost is lower. Detection rate (DR) is defined as follows:

$$DR = \frac{\text{captured attackers}}{\text{all the attackers}}. \quad (12)$$

4.3. Experiment Results. Firstly, the correctness of the method proposed in this paper is verified. In this experiment, we want to prove that the proposed model can discover hidden attack sequences, and also, this experiment proves that the cyberspace has hidden attack sequences. It indicates that the attacker's hidden attack sequences exist in this cyberspace security configuration. In this experiment,

when the value of R_{cost} is set to 1–20, the DDPG model is trained 500 times, and the training process is recorded and visualized. We choose $R_{\text{cost}} = 10$; the changes in the overall reward value are shown in Figure 7. This is Experiment A.

Secondly, the effects of different parameters on the performance of the model are further compared. The model is trained when the value of R_{cost} changes from 1 to 20, R_c changes from 50 to 240, R_s changes from -240 to -50, and C changes from 100 to 500. Compare the average reward of the models under different parameters. We choose R_{cost} result as shown in Figure 8. This is Experiment B.

Finally, the superiority of the proposed method is verified, and this is Experiment C. In the same environment, these methods will be compared. In the experiment setting, DRDDPG reward setting as shown in equation (8), in other methods, we set that if an agent catch attacker, the agent will get a reward value of 100; if the attacker attacks successfully, agent will get a reward value of -100; if the agent checks cyberspace but no harvest, we consider agent will cost cyberspace management costs, and the agent will get a reward value of R_{cost} . From Experiment B, we can see that when

FIGURE 8: Parameter experiment results: the y -axis is the total reward agent obtained in an episode and the x -axis is parameters in method: R_{cost} is the parameter when the agent checks cyberspace but no harvest; R_c is the parameter when the attacker is captured; R_s is the parameter when attacked successfully; C is a constant term, which makes sure that R does not change dramatically.

$R_{cost} = 10$, the reward value is better, and considering other parameters' values, we finally set $R_{cost} = 10$. In other cases, agent will get a reward of 0.

In the random method, we set that γ represents the proportion of agent's check cyberspace actions in the total actions. Above that, we set two methods. One is Random (0-1), which means γ is set to 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1. $\gamma = 1$ means agent check cyberspace each time, and the result is that all attacker will be captured, but the agent will cost high cyberspace management cost. The other is Random (0.5), which means that γ is set to 0.5.

In DRDDPG, its parameters will be discussed. From the discussion results, we set the parameters $R_{cost} = 10$, $R_c = 100$, $R_s = -100$, and $C = 200$.

In Experiment C, all the methods were tested 11 times, and the results are shown in Table 1. In these experiments, except the Random (0-1) method, γ is different; other same method experiments are done with the same parameter settings.

4.4. Discussion. Firstly, we see Experiment A results. Through the results of Figure 7, it can be found that, with the increasing number of training times, the reward value shows a slowly rising trend until it finally approaches convergence, which is consistent with a learning process of reinforcement learning, indicating that the model proposed in this paper can monitor user's action and then gradually learns the characteristics of attacker's actions, and their judgment accuracy is continuously improved, which proves the effectiveness of the proposed model.

Secondly, we see Experiment B results. Through the results of Figure 8, it can be found that, no matter how the administrator changes R_{cost} in this environment, the proposed method can always get a better reward, which also means that the method we proposed can adapt well to the cyberspace environment and adjust its policy according to changes in the environment, furthermore, the model can not only achieve better results in a specific environment but also

TABLE 1: Experiment results.

Method	Reward	Captured attackers	Successful attackers	Detection rate	Cost reduction
Packet detection	5455.6 $\pm$ 572.28	145.6 $\pm$ 20.78	19.6 $\pm$ 8.77	87.77%	$\times$
Random (0-1)	-12 455.6 $\pm$ 9528.77	127.36 $\pm$ 26.64	26.41 $\pm$ 48.31	82.82%	$\times$
Random (0.5)	-4146.2 $\pm$ 752.7	148.8 $\pm$ 15.46	15.32 $\pm$ 10.58	90.6%	$\times$
DQN	11 461.82 $\pm$ 472.39	151.66 $\pm$ 5.53	12.8 $\pm$ 4.8	92.21%	$\times$
DDPG	13 664.45 $\pm$ 452.8	155.45 $\pm$ 2.54	8.18 $\pm$ 3.68	95%	19.21% $\rightarrow$ DQN
DRDDPG	14 614.13 $\pm$ 433.3	157.48 $\pm$ 3.72	4.1 $\pm$ 2.01	97.46%	6% $\rightarrow$ DDPG

achieve better results in different environments for this problem, proving that the model has good robustness. Also, we can find from experiment results that, with R_s and R_c getting higher and higher, the reward value is becoming higher and higher, and this is consistent with our general understanding. The higher the individual reward you give to an agent, the higher the cumulative reward will be. But if R_s and R_c are set higher, the parameter of reward in a contrastive experiment will cause loss of fairness. Therefore, we will try to be consistent with the reward settings in other methods to ensure fairness, and we set $R_s = -100$, $R_c = 100$, $R_{cost} = 10$, and $C = 50$.

Finally, through the results of Table 1, we can find that our proposed method performs better than the existing methods, regardless of the reward obtained, the number of attackers captured, and the number of attackers who have successfully attacked the server. Therefore, the method we proposed can achieve better rewards while discovering more attackers, which means that our proposed method can discover more attackers with less cost.

From the results of Table 1, it can be found that, for the random check method, when the proportion of the check action is low, it means that the agent does not check cyberspace regardless of the cyberspace state, resulting in the fact that the number of attackers successfully attacked is high, and the number of attackers captured is little, resulting in a lower average reward value. When the proportion of checking actions gradually increases, the agent can successfully catch more attackers, resulting in a gradual increase in the average value of the reward. But when the checking action proportion is more, the agent is equivalent to wasting most of the time in checking cyberspace. At this time, although more attackers can be found, a lot of costs are wasted. Due to the existence of R_{cost} , the reward at this time is gradually decreasing; therefore, the variance of Random (0-1) results is much greater than other methods. At the same time, its mean is far less than other methods, and the reasons are the same as above.

In this experiment, the random method's reward is minimum because the agent checks the cyberspace's states frequently, due to R_{cost} , and the agent lost many rewards, resulting in low rewards. Also, the captured attackers are not much different. This may be an attacker launching an attack, though he has many hidden attack paths to launch an attack, but he will leave a trail in cyberspace. So the possibility of getting caught is pretty good. But in successful attackers, the random method is higher. The reason may be the agent randomly catches attackers. With our proposed methods, the agent can catch more attackers; the reason may be our

methods can judge attackers or normal users by cyberspace's state so as to catch more attackers, not normal users, even though there are many normal users among attackers.

From the experimental results, the random method's DR is low and costs much cyberspace management cost because its reward is low. At the same time, three methods based on reinforcement learning have a better performance, which proves that our model based on reinforcement learning can discover the attacker who attacks by hidden sequences.

5. Conclusion

In daily network security management, most attackers' behaviors are identified and discovered by capturing and analyzing the target network traffic. It is almost impossible to detect all hidden attacks through monitoring all network traffic due to the unbearable cost of equipment and management, which has developed into a typical dilemma. And the hidden sequence attack is the key point of the dilemma. The present paper built a typical cyberspace environment in which the attackers can bypass the detection under the shield of legitimate actions and perform attacks, and we called it hidden attack sequences. Correspondingly, this paper proposed a hidden attack sequence detection method based on reinforcement learning. In this method, the administrator is modeled as the agent, which chooses their actions based on the sensed cyberspace state and DRDDPG method. With this method, the administrator could identify the attacker who bypasses the security detection device and attacks through legal actions, decrease the cyberspace management cost, and achieve a better balance between improving the ability to discover hidden attackers and reducing cyberspace management costs. The experimental results certificated that the proposed method can improve the ability of administrators to discover hidden attack sequences and to reduce cyberspace management costs in a typical cyberspace environment.

This paper analyzes a typical cyberspace environment and applies the reinforcement learning method to network management action, which has achieved better results. However, the cyberspace environment in this paper is limited. In the next step, we hope to apply this method to more network operation and maintenance management and achieve better results.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 62076251) and grants from the National Key Research and Development Program of China (no. 2017YFB0802800).

References

- [1] Y. B. Zhang, Z. B. Zhang, Y. Zhao, and G. Li, "Comparative analysis of TCP and UDP network traffic," *Application Research of Computers*, vol. 27, no. 6, 2010.
- [2] H. J. Liao, C. H. Richard Lin, Y. C. Lin, and K. Y. Tung, "Intrusion detection system: a comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.
- [3] Y. P. Jiang, C. C. Cao, and M. Xiao, "Research progress and prospect of network intrusion detection technology," *Journal of Light Industry*, vol. 32, 2017.
- [4] X. Zhang, C. Li, and W. Zheng, "Intrusion prevention system design," in *Proceedings of the International Conference on Computer and Information Technology*, Wuhan, China, September 2004.
- [5] A. Patel, M. Taghavi, K. Bakhtiyari, and J. Celestino Júnior, "An intrusion detection and prevention system in cloud computing: a systematic review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 25–41, 2013.
- [6] J. Gong, X. D. Zang, S. U. Qi, X. Y. Hu, and X. U. Jie, "Survey of network security situation awareness," *Journal of Software*, vol. 28, 2017.
- [7] L. Zhang, W. Bai, S. Guo, S. Xia, H. Li, and Z. Pan, "Weakness analysis of cyberspace configuration based on reinforcement learning," 2020, <https://arxiv.org/abs/2007.04614>, Article ID 04614.
- [8] R. Samrin and D. Vasumathi, "Review on anomaly based network intrusion detection system," in *Proceedings of the International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques*, Mysuru, India, December 2017.
- [9] W. Bulrajoul, A. James, and S. Shaikh, "A new architecture for network intrusion detection and prevention," *IEEE Access*, vol. 7, 2019.
- [10] A. Dastanpour, S. Ibrahim, R. Mashinchi, and A. Selamat, "Comparison of genetic algorithm optimization on artificial neural network and support vector machine in intrusion detection system," in *Proceedings of the 2014 IEEE Conference on Open Systems*, Subang, Malaysia, October 2014.
- [11] N. Gao, L. Gao, Y. Y. He, and H. Wang, "A lightweight intrusion detection model based on autoencoder network with feature reduction," *Acta Electronica Sinica*, vol. 45, 2017.
- [12] B. Xu, S. Chen, H. Zhang, and T. Wu, "Incremental k-NN SVM method in intrusion detection," in *Proceedings of the 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China, November 2017.
- [13] B. Selvakumar and K. Muneeswaran, "Firefly algorithm based feature selection for network intrusion detection," *Computers & Security*, vol. 81, pp. 148–155, 2019.
- [14] Akashdeep, I. Manzoor, and N. Kumar, "A feature reduced intrusion detection system using ANN classifier," *Expert Systems with Applications*, vol. 88, pp. 249–257, 2017.
- [15] G. Ni, G. Ling, Q. Gao, and W. Hai, "An intrusion detection model based on deep belief networks," in *Proceedings of the Second International Conference on Advanced Cloud and Big Data*, Huangshan, China, November 2014.
- [16] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, 2018.
- [17] C. L. Yin, Y. F. Zhu, J. L. Fei, and X. Z. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, no. 99, p. 1, 2017.
- [18] T. Le, J. Kim, and H. Kim, "An effective intrusion detection classifier using long short-term memory with gradient descent optimization," in *Proceedings of the International Conference on Platform Technology and Service*, Busan, Republic of Korea, February 2017.
- [19] N. Sheraz, S. Yasir, K. Shehzad et al., "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, 2018.
- [20] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, "Variational LSTM enhanced anomaly detection for industrial big data," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3469–3477, 2021.
- [21] M. G. Lagoudakis and R. Parr, "Reinforcement learning as classification: leveraging large margin classifiers," *ICML Submitted*, 2003.
- [22] A. Servin, "Towards traffic anomaly detection via reinforcement learning and data flow," *IEEE Access*, vol. 6, 2018.
- [23] D. K. Y. Yau, J. C. S. Lui, F. Liang, and Y. Yam, "Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles," *IEEE/ACM Transactions on Networking*, vol. 13, no. 1, pp. 29–42, 2005.
- [24] M. Zhu, Z. Hu, and P. Liu, "Reinforcement learning algorithms for adaptive cyber defense against Heartbleed," in *Proceedings of the First ACM Workshop on Moving Target Defense*, pp. 51–58, Scottsdale AZ USA, November 2014.
- [25] R. Elderman, L. J. J. Pater, A. S. Thie, M. M. Drugan, and M. A. Wiering, "Adversarial reinforcement learning in a cyber security simulation," in *Proceedings of the International Conference on Agents and Artificial Intelligence*, January 2017.
- [26] G. Caminero, M. Lopez-Martin, and B. Carro, "Adversarial environment reinforcement learning algorithm for intrusion detection," *Computer Networks*, vol. 159, pp. 96–109, 2019.
- [27] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 1–17, 2021.
- [28] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction* MIT Press, London UK, 1998.
- [29] C. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, no. 3–4, pp. 279–292, 1992.
- [30] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Playing atari with deep reinforcement learning," *Computer Science*, 2013, <https://arxiv.org/abs/1312.5602>.
- [31] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the International conference on machine learning*, pp. 387–395, PMLR, Beijing China, June 2014.
- [32] T. P. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," 2015, <https://arxiv.org/abs/1509.02971>.